# UNIT 5: INSTRUCTION SETS

## UNIT STRUCTURE


5.1 Learning Objectives

5.2 Introduction

5.3 Elements of a Machine Instruction

5.4 Instruction Representation

5.5 Instruction Types

5.6 Number of Addresses

5.7 Types of Operands

5.8 Types of Operations

- 5.8.1 Data Transfer Operation
- 5.8.2 Data Processing Operation
- 5.8.3 Program Sequencing and Control Operation

5.9 Instruction Formats

5.10 Assembly Language Notation

5.11 Let Us Sum Up

5.12 Further Reading

5.13 Answers to Check Your Progress

5.14 Model Questions


## 5.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- describe the elements of machine instructions
- learn about instruction format, representation and types of instructions
- explain different types of operands and operations
- learn about assembly language notations.

## 5.2 INTRODUCTION

In the previous unit we have learnt the numbers that are represented in the binary number system and the 2's complement system. We have already discussed the fundamental issues related to the arithmetic operations

used to support computations in computer. In this unit, we will focus on the way the programs are executed in a computer from the machine instruction viewpoint.

The program instructions and the data operands are stored in the memory. The instructions are brought from the memory to the processor and then executed to perform a given task. Operands and results must also be moved between the memory and the processor. The tasks carried out by the computer consist of some small steps. The computer must have instructions capable of performing the different types of operations. In this unit, we will discuss about the machine instructions and the different types.

## 5.3 ELEMENTS OF A MACHINE INSTRUCTION

The operation of a computer is determined by the instruction executed by the CPU. Such instructions are referred to as machine instructions or computer instructions. A collection of such instructions that the computer can execute is called an instruction set.

There are different elements of a machine instruction. These are as follows:

1) Operation Code.

   The operation code specifies the operation to be performed. For example, ADD, MOV, MULT etc. The operation is specified by a binary code.

2) Source operand reference.

   Source operand references are the operands that are the inputs for the operation. i.e., operands on which operation is to be performed.

3) Result operand reference.

   The operation may produce a result.

4) Next instruction reference.

After the execution of an instruction is complete, the Next instruction reference tells where from the next instruction is to be fetched.

Generally, the next instruction immediately follows the current instruction and there is no explicit reference to the next instruction. However, in some cases, the next instruction may be in the main memory or in the

secondary memory. In such cases, there is an explicit reference to the next instruction, and then the main memory or virtual memory address must be supplied.

## 5.4 INSTRUCTION REPRESENTATION

Each of the machine instruction is represented by a sequence of bits in the computer. The instructions are divided into fields corresponding to the constituent elements of the instruction. The machine instructions are highly machine specific.

An example of an instruction is shown in the figure below. Let us assume that the CPU is a 16-bit CPU. 4-bits are used for the operation code. Thus there can be $2^4 = 16$ different sets of instructions. The instruction has two operands each of 6-bits. Thus, there can be $2^6 = 64$ different operands for each operand reference.

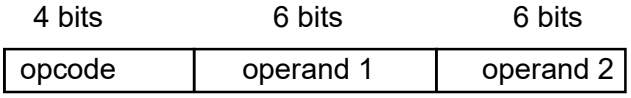



**Figure. 5.1: A Simple Instruction Format**

To deal with binary representation is not an easy task and so a symbolic representation of the machine instructions is used. The opcodes are represented by abbreviations called *mnemonics*. Common examples are:

| | |
|---|---|
| *ADD* | Add |
| *SUB* | Subtract |
| *MULT* | Multiply |
| *DIV* | Division |
| *LOAD* | Load data from memory to CPU |
| *STORE* | Store data to memory from CPU |

Let us take the following machine instruction as an example,

**ADD R, A**

This instruction means add the value contained in memory location A to the contents of the register R.

## 5.5 INSTRUCTION TYPES

A computer should have a set of instructions that allows the users to formulate any data processing works. Instructions can be classified according to their operations and address reference.

According to the operations, the instructions can be classified as

a) Data Processing
b) Data Storage
c) Data Movement
d) Control

*Data Processing*: The data processing instructions include both logic and arithmetic instructions. The logic instructions perform the logical operations on the bits of the word. The arithmetic instructions provide computational capabilities for processing numeric data.

*Data Storage:* The logical and arithmetic operations are performed on the data in the processor registers. So there must be memory instructions for transferring the data or moving the data from the memory to the processor registers before operation and transfer data from CPU registers to memory after operations.

*Data movement:* The I/O instructions constitute the data movement instructions. These instructions are needed to transfer the data and the programs between the memory registers and input-output devices.

*Control:* The test and branch instructions are the control instructions. The test instructions test the value of a word or the status of a computation. The branch instructions are used to branch to a different set of instructions.

## 5.6 NUMBER OF ADDRESSES

The processor architecture can also be described in terms of the number of addresses contained in each instruction. According to the address reference, the instructions can be classified as three address instructions, two address instructions, one address instructions and zero address instructions.

*Three address instruction:* Symbolically, the three address instruction can be represented as follows:

ADD A, B, C

Where A, B and C are the variables. A and B are called Source operands and variable C is called destination operand. ADD is the operation to be performed on the operands.

*Two address instruction:* Symbolically, it can be represented as follows:

ADD A, B

This instruction adds up the contents in A and B and stores the resultant in B overwriting the previous contents of B. Here, A is the source operand and B is both source as well as destination operand.

*One address instruction:* Symbolically, it can be represented as follows:

ADD A

This instruction adds the contents of A to the contents of the processor register called accumulator and then the result is stored back to the accumulator destroying the previous contents. Another example of a one address instruction is:

LOAD A

It copies the contents of the memory location A into the accumulator.

*Zero address instruction:* These types of instructions are used in machines that stores operands in a pushdown stack. A stack stores the locations in last-in-first-out order.



**CHECK YOUR PROGRESS**

**Q.1:** Fill in the blanks:

i) The ..... are directly executed by the CPU of a computer.

ii) The ..... specifies the operation to be performed.

iii) Machine instruction is represented by a sequence of ..... in the computer.

iv) Instructions can be classified according to their operations and .....

v) The arithmetic instructions provide ..... capabilities for processing numeric data.

vi) According to the address reference, the machine instructions can be classified as ....., ....., ..... and ..... instructions.

vii) The ..... and ..... instructions are the control instructions.

## 5.7 TYPES OF OPERANDS

Every instruction has two parts– *operands* and *operation code* (opcode). Operand is another name for data. There may be different types of operands. There may be different types of operands:

a) Addresses
b) Numbers
c) Characters
d) Logical data

*Addresses:* sometimes, calculations must be performed on the operand reference in an instruction to determine physical address.

*Numbers:* The common numeric data types are integer, point and decimal.

*Characters:* The common form of data for documentation purpose is character strings. Most of the computers use ASCII (American Standard Code for Information Interchange) code for characters represented by 7-bit pattern and 8-bit patterns. The $8^{th}$ bit may be set to zero or used as a parity bit for error detection. The EBCDIC (Extended Binary Coded Decimal Interchange Code) code is also used to encode characters.

*Logical data:* The units of data are bit, byte, word or double word. Bit means either a 0 or 1. When data is viewed as a 1-bit data, it is logical data. The logical data are used to store Boolean or binary data items. Using logical data, the bits of data can be manipulated.

## 5.8 TYPES OF OPERATIONS

A set of instructions are given to the computer to carry out different data processing tasks. The following are the basic types of operations that the machine instructions should be able to carry out.

- Data Transfer
- Data Processing
- Program Sequencing and Control

### 5.8.1 Data Transfer Operations

The data transfer operations are the most fundamental type of operations. Such transfer operations that are included here are as given below.

a) Data transfer between memory and CPU registers
b) Data transfers between CPU registers
c) Data transfer between processor and input-output devices

These instructions must specify the location of the source and the destination operands. Also the length of the data to be transferred and the mode of addressing must also be specified.

**a) Data transfer between memory and CPU registers:** The arithmetic and logical operations are performed on the data stored in the processor registers. Thus, before any operation, the data needs to be transferred from the memory to the CPU registers and after the operation, the resultant data has to be transferred from those registers to the memory. The two basic operations that are involved in these memory access are **Load** (Read) and **Store** (Write).

***LOAD*** operation fetches word from memory to the processor. **STORE** operation transfers the data from the processor to the memory. The data or instructions which are transferred between processor and memory may be a byte or a word. The processor has a number of registers.

Example, R2 ⟵ [LOC]

This instruction transfers the contents of memory location to the register R2.

b) **Data transfers between CPU registers:** Data transfers also occur between the CPU registers. If the source and destination are registers, then the processor simply causes data to be transferred from one register to another. This operation is internal to the processor. As for example, R3 ⟵ [R2]

This instruction transfers data from CPU register R2 to the CPU register R3.

c) **Data transfer between processor and input-output devices:** The transfer of data between the input devices and processor and the output devices is called I/O data transfer. For example, R1 ⟵ [DATA IN]

This instruction states that the contents of I/O register, DATA IN are transferred to the processor register R1. DATA IN and DATA OUT are the registers by which the processor reads the contents from the keyboard and sends the data for display.

The rate of output data transfer is much higher than the rate of the input data transfer. However, the speed of a processor is much higher than these and can execute millions of instructions per second. So, to overcome the speed difference between these devices, synchronization mechanism should be used for proper transfer of data between them.

As shown in the figure below, the $\mathbf{S_{IN}}$ and $\mathbf{S_{OUT}}$ status bits are used to synchronize data transfer between display and the processor respectively.

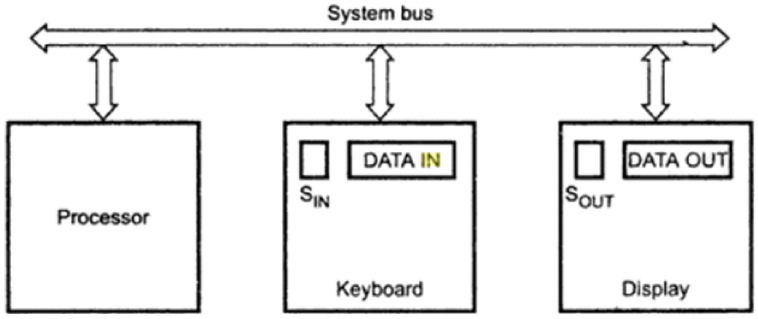



**Figure 5.2: Bus Connection for Processor, Keyboard and Display**

When a key is pressed, the corresponding character code is stored in DATA IN and the $S_{IN}$ bit is set to 1. When the processor finds the $S_{IN}$ = 1, it reads the contents of DATA IN register. After completing the read operation, the $S_{IN}$ is automatically set to 0. If another key is pressed, the corresponding character is entered in the DATA IN register and the $S_{IN}$ is set to 1 and the process is repeated again.

Similarly, when a character is to be transferred from processor to the display, the DATA OUT register and the $S_{OUT}$ status bits are used. When the processor wants to transfer data to the display unit, the $S_{OUT}$ is set to 1 and after the transfer is completed the $S_{OUT}$ bit is cleared to 0.

### 5.8.2 Data Processing Operations

Data processing operations include the following types of operations.

- Arithmetic operations,
- Logical operations,
- Shift and rotate operations

The arithmetic operations basically include the following:

Add,
Subtract,
Multiply,
Divide,
Increment,
Decrement,
Negate.

For example, to add the contents of the registers R1 and R2, and to store the results in another register R3, the following instruction is used.

**ADD R1, R2, R3**

The logical operations include the following:

AND,
OR,

NOT,

XOR,

Compare,

Shift,

Rotate

For example, the expression:

**AND R1, R2, R3**

States that the contents of the processor registers R1 and R2 are logically AND operated and the result is stored in R3.

The **Compare** operation makes logical or arithmetic comparisons of two or more operands. With a logical shift, the bits of a word are shifted left or right. There are two logical shift instructions: logical shift left and logical shift right.



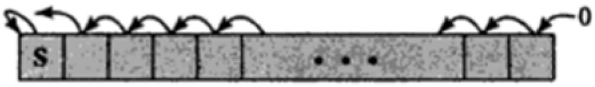
(a) Logical Right Shift



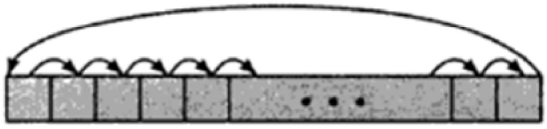

(b) Logical Left Shift



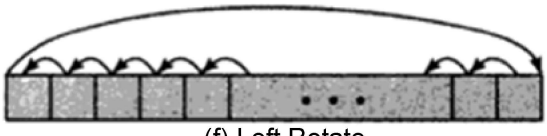

(c) Arithmetic Right Shift




(d) Arithmetic Left Shift



(e) Right Rotate



(f) Left Rotate

**Figure 5.3: Shift and Rotate Operation**

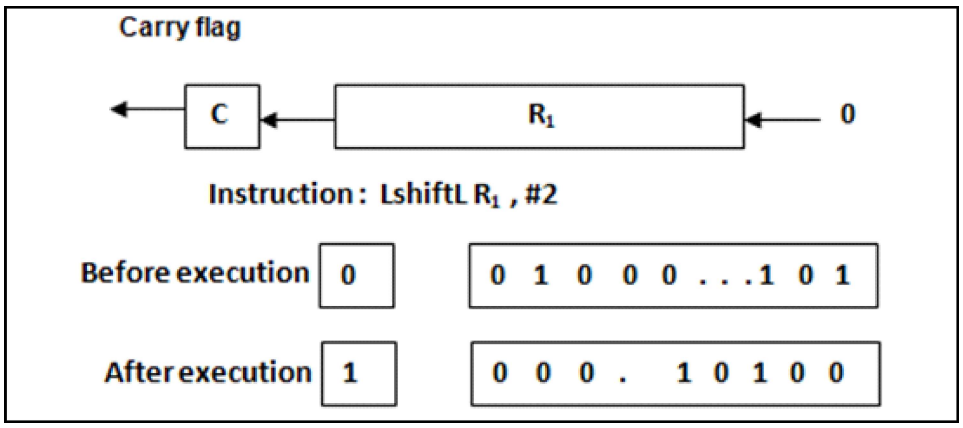
On one end, the bit shifted out is lost. On the other end, a 0 is shifted in. The logical shifts are used for isolating fields within a word. The 0s that are shifted into the word displace the unwanted information which is shifted off from the other end.

The general syntax for logical shift operations are:

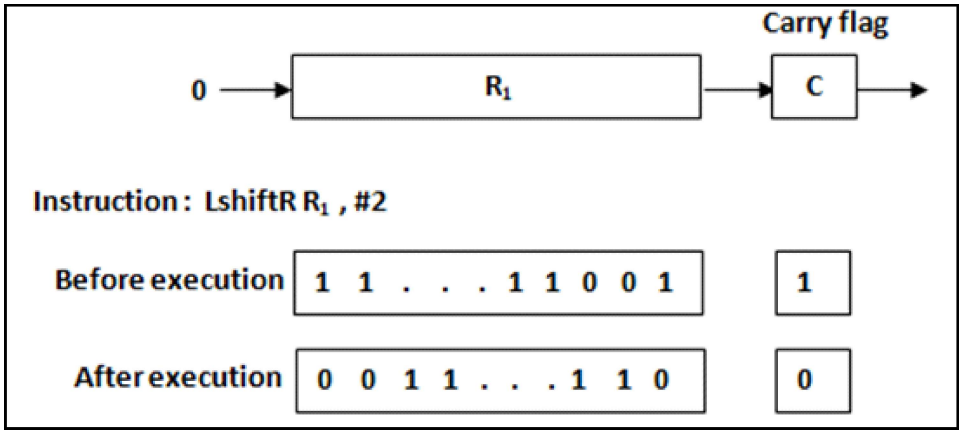
LshiftL dst, count

LshiftR dst, count

The count may be a number or the contents of a processor register. The figure (Figure. 5.4) shows the operation of LshiftL and LshiftR operations.




(a) Example of LShiftL instruction




(b) Example of LShiftR instruction

**Figure: 5.4 : LshiftL and LshiftR operations**

***Rotate*** is also called cyclic shift operation. In shift operations, the bits shifted out of the operand are lost except for the last bit which is retained in the Carry flag (C). On the other hand, the rotate operation

preserves all the bits being operated on. One of the uses of the rotate operation is to bring each bit successively into the leftmost bit, where it can be identified by testing the sign of the data. There are two basic types of rotate operations– rotate left and rotate right.

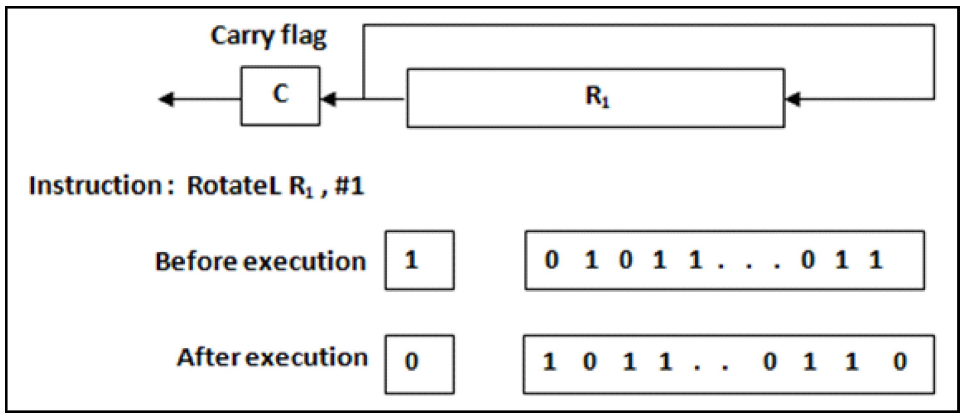



**Figure: 5.5: Example of RotateL Instruction**

### 5.8.3 Program Sequencing and Control

The program sequencing and control mainly includes the test and branch operations. Test operations are used to test the value of data and the branch operations are used to branch to a different set of operations depending on the decision made. Programs sometimes need decision making. The computer is asked to perform one thing if one condition is met and another thing if another condition holds.

A branch instruction is also called a jump instruction. One of the operand of a branch instruction is the address of the next instruction to be executed. Most often, the instruction is a conditional branch instruction. That is, a jump is made only if a condition is met otherwise the next instruction in the sequence is executed. If in a branch instruction, the branch is always made, then it is an unconditional branch.

A branch can be either a forward (an instruction with a higher address) or a backward (an instruction with a lower address). The figure below shows how an unconditional and a conditional branch

can be used to create a repeating loop of instructions. The instructions in locations 202 through 210 will be executed repeatedly until the result of subtracting Y from X is 0.

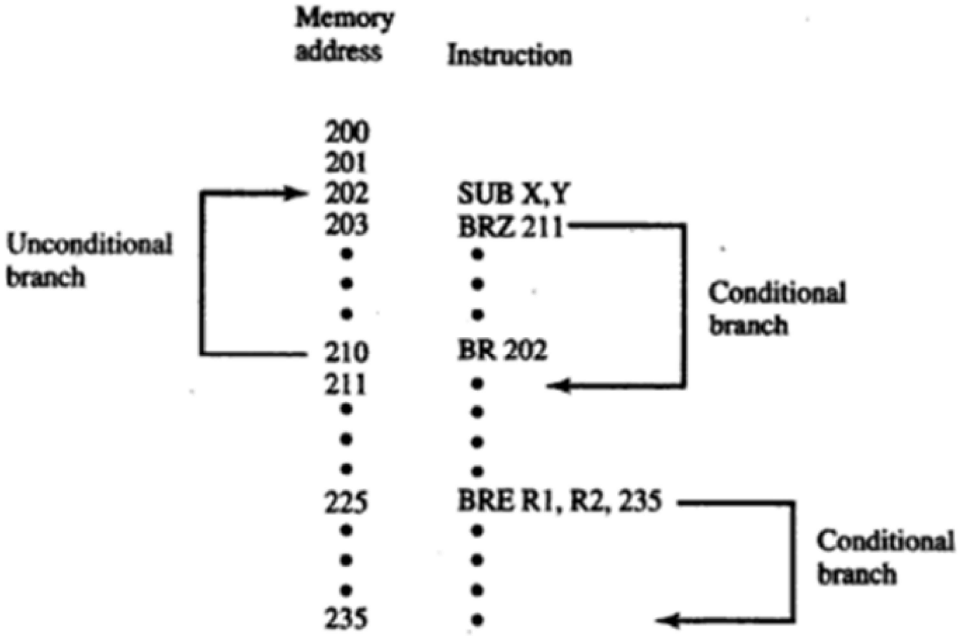



**Figure.5.6: Branch Instructions**

Skip instructions are also transfer of control instructions. The skip implies that an instruction be skipped and thus the implied address of the skip instruction is equal to the address of the next instruction plus one instruction length.

## 5.9 INSTRUCTION FORMATS

The CPU fetches the instruction from the memory, decodes it and executes the instruction according to its type. The structure of an instruction is called instruction format. It defines the layout of the bits of an instruction in terms of its constituent fields. The instruction format consists of an operation code or opcode, and zero or more operands. The operands may be implicit or explicit. The explicit operands are referenced using one of the addressing modes discussed next.

An instruction is of various lengths depending upon the number of addresses it contains. The design of an instruction format is complex. The key design issues are discussed below.

***Instruction length:*** The instruction length is affected by the memory size, organization, bus structure, processor speed and processor complexity. The programmers want more opcodes, more operands, more addressing modes and greater address ranges. Shorter programs can be written to do a particular task with more opcodes and more operands. More addressing modes also gives the programmers flexibility in implementing certain functions. These requirements of more number of things requires more bits and so results in longer instructions. However, sometimes longer instructions may be not that useful and be a waste of space.

There are other considerations also beyond the length. Either the length of the instruction should be equal to the memory transfer length or one should be a multiple of the other. A related consideration is the memory transfer rate. The processor speed is very fast and can execute more number of instructions than the number that is fetched from the memory. So a solution to this can be to use a cache memory, another solution can be to use shorter instructions. Thus 16-bit instructions can be fetched at twice the rate of 32-bit instructions but probably can be executed less than twice as fast.

***Allocation of bits:*** Another difficult issue with the design of the instruction format is how to allocate the bits in that format. The trade-offs are complex. There are some factors that can be useful to determine the use of addressing bits.

- Number of addressing modes. Addressing modes may be implicit or explicit.
- Number of operands
- Register versus memory. With registers only a few bits are needed to specify the register. The more registers can be used for operand references, the fewer bits are needed.
- Address range. The range of addresses that can reference memory is related to the number of address bits. As this has a severe limitation, direct addressing is rarely used. With displacement addressing, the range is opened up to the length of the address register.
- Address granularity. Granularity of addresses is another important factor for addresses that references memory rather than registers.

## 5.10 ASSEMBLY LANGUAGE NOTATION

Assembly language notations are used to represent machine instructions and programs. These notations use assembly language formats. Example of some assembly notations are as follows.

**MOVE R1, R2**

This assembly language instruction involves the transfer of contents between the registers. The contents of register R1 are transferred to R2. The contents of R1 are unchanged but those of R2 are overwritten.

**ADD R1, R2, R3**

This expression states that the contents of the registers R1 and R2 are added and the result is stored in the register R3.

The assembly language notations have three fields– operation, source and destination.



### CHECK YOUR PROGRESS

**Q.2:** Fill in the blanks:

i) ..... and ..... are the two parts of an instruction.

ii) When data is viewed as a ..... data, it is logical data.

iii) Logical data are used to store ..... items.

iv) The ..... and ..... operations are performed on the data stored in the processor registers.

v) The two basic operations that are involved in memory access are ..... and .....

vi) ..... Operation transfers the data from the processor to the memory.

vii) The transfer of data between the input devices and processor and the output devices is called ..... transfer.

viii) There are two logical shift instructions called ..... and .....

ix) Rotate is also called ..... operation.

x) A branch instruction is also called a ..... instruction.



## 5.11 LET US SUM UP

- The operation of a computer is determined by the instruction executed by the CPU. A collection of such instructions that the computer can execute is called an *instruction set*.
- Each of the machine instruction is represented by a sequence of bits in the computer.
- The opcodes are represented by abbreviations called *mnemonics*.
- The machine instructions are represented by a sequence of bits in the computer.
- Instructions can be classified according to their operations and address reference.
- According to the operations, the instructions can be classified as Data Processing, Data Storage, Data Movement, and Control.
- According to the address reference, the instructions can be classified as three address instructions, two address instructions, one address instructions and zero address instructions.
- Every instruction has two parts– operands and operation code (opcode).
- The basic types of operations that the machine instructions should be able to carry out are Data transfer, Data processing and Program sequencing and control.
- Data transfer operations may be:
  - Data transfer between memory and CPU registers,
  - Data transfers between CPU registers,
  - Data transfer between processor and input-output devices.
- The two basic operations that are involved in memory access are Load (Read) and Store (Write).

- Data processing operations includes,
  - Arithmetic operations,
  - Logical operations,
  - Shift and rotate operations.
- The *Compare* operation makes logical or arithmetic comparisons of two or more operands.
- With a logical shift, the bits of a word are shifted left or right.
- Rotate is also called cyclic shift operation.
- In shift operations, the bits shifted out of the operand are lost except for the last bit which is retained in the Carry flag (C).
- Rotate operation preserves all the bits being operated on.
- Test operations are used to test the value of data and the branch operations are used to branch to a different set of operations depending on the decision made.
- A branch instruction is also called a jump instruction. One of the operand of a branch instruction is the address of the next instruction to be executed.
- A branch can be either a forward (an instruction with a higher address) or a backward (an instruction with a lower address).
- Skip instructions are also transfer of control instructions.
- The structure of an instruction is called instruction format. It defines the layout of the bits of an instruction in terms of its constituent fields.
- Assembly language notations are used to represent machine instructions and programs.
- The assembly language notations have three fields namely operation, source and destination.



## 5.12 FURTHER READING


1) Chaudhuri, P. Pal. (2nd Edition, 2003). *Compter Organization and Design*. PHI Learning Pvt. Ltd.
2) Gill, D. N. S. & Dixit, J. B. (2008). *Digital Design and Computer Organisation*. Firewall Media.

3) Null, L. & Lobur, J. (2014). *The Essentials of Computer Organization and Architecture*. Jones & Bartlett Publishers.

4) Stallings, William. (2004). *Computer Organization and Architecture Designing for Performance*. Pearson Education India.



## 5.13 ANSWERS TO CHECK YOUR PROGRESS

**Ans. to Q. No. 1:** i) Machine instructions, ii) operation code, iii) bits, iv) address reference, v) computational, vi) three address, two address, one address, zero address, vii) test, branch

**Ans. to Q. No. 2:** i) Operands, operation code; ii) 1-bit; iii) Boolean data; iv) Arithmetic, logical; v) Read, Write; vi) Store; vii) I/O data; viii) logical shift left, logical shift right; ix) cyclic shift; x) jump



## 5.14 MODEL QUESTIONS

**Q.1:** What are the elements of a machine instruction? Explain.

**Q.2:** How can a machine instruction be represented?

**Q.3:** Explain the various types of operands and operations.

**Q.4:** What are Shift and Rotate logical operations?

**Q.5:** Describe the Assembly language notation.

*** ***** ***

# UNIT 6: ADDRESSING MODES

## UNIT STRUCTURE


6.1 Learning Objectives

6.2 Introduction

6.3 Addressing

6.4 Immediate Addressing

6.5 Direct Addressing

6.6 Indirect Addressing

6.7 Register

6.8 Register Indirect

6.9 Relative Index

6.10 Let Us Sum Up

6.11 Further Reading

6.12 Answers to Check Your Progress

6.13 Model Questions


## 6.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn about addressing
- describe how addressing is done
- describe different addressing modes for accessing register and memory operands.

## 6.2 INTRODUCTION

In the previous unit we have learnt what the machine instructions does and their formats and types. We have examined the type of operands and operations that may be specified by machine instructions. Also we have gone through the assembly language notation. In this unit, we will learn about how the address of an operand is specified, and how are the bits of an instruction organized to define the operand addresses and operation of that instruction.

An instruction is a group of bits that instruct the computer to perform a specific operation. A sequence of instructions is called a computer program. An instruction specifies the operation in the operation code field on the data specified by the operands in the operand field. There are different types of addressing capabilities in an instruction format. Addressing is an instruction design issue. There are different addressing modes that allow us to specify where the operands are located. An addressing mode can specify a location in memory, or a register or a constant. There are certain modes that allow shorter addresses, and some allow us to determine the location of the actual operand, also called the effective address of the operand dynamically. The typical addressing modes that are employed in the processor of a computer are discussed in this unit.

## 6.3 ADDRESSING

The techniques of specifying the address of the data are known as addressing modes. Thus the term addressing refers to the way in which the operand of an instruction is specified. The addressing mode specifies a rule for interpreting the address field of an instruction before the instruction is actually executed. Computers use addressing mode techniques for the following purposes:

a) To reduce the number of bits in the field of instruction,
b) Giving programming versatility to the user by providing facilities such as pointers to memory
c) Specifying rules for modifying or interpreting address field of the instruction.

There are a variety of addressing techniques that are been applied. The most common of them are:

- Immediate
- Direct
- Indirect
- Register
- Register Indirect

The computer architectures provide more than one of these addressing modes. So, there is a question that how the processor can determine which addressing mode is being used in a particular instruction. There are several approaches for this to be done. Often, the different symbols/assembler syntax are used for different addressing modes. Also, one or more bits in the instruction format can be used as a *mode* field.

To explain the addressing modes, we use the following notation:

A = contents of an address field in the instruction that refers to a memory

R = contents of an address field in the instruction that refers to a register

EA = actual (effective) address of the location containing the referenced operand

(X) = contents of location X

The different addressing modes are explained in details in the sections below. Let us take an example to understand the various addressing modes. Suppose the two word instruction “LOAD AC”. The instruction is at

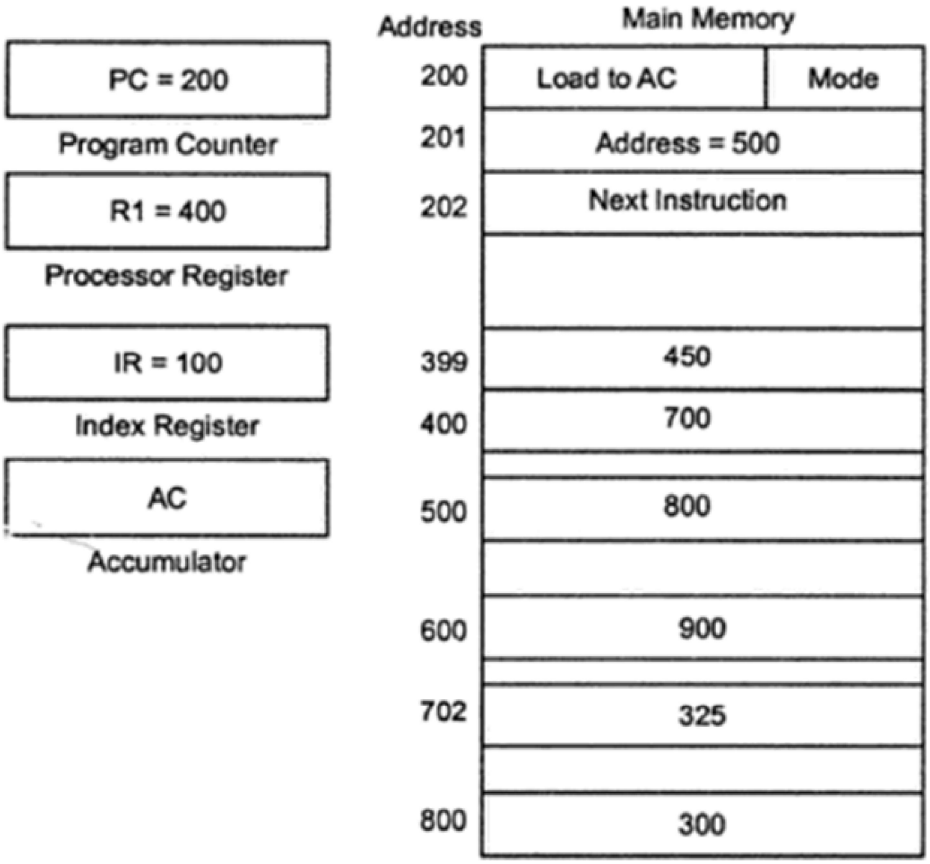



**Figure 6.1. Example for Addressing Mode**

address 200 and 201. The address field is 500. The first word of the instruction specifies the operation code and mode. The second word specifies the address part. The PC (Program Counter) which fetches the next instruction has the value 200. The content of the processor register R1 is suppose 400 and that of the Index register is 100. The Accumulator, AC receives the operand after the instruction is executed.

## 6.4 IMMEDIATE ADDRESSING

This is the simplest form of addressing. In the immediate mode, the operand is specified in the instruction itself. An immediate mode instruction has an operand field rather than the address field. The operand can be a byte, word or double word of data. Immediate addressing is so-named because the value to be referenced immediately follows the operation code in the instruction. Thus, the data to be operated on is part of the instruction.

**OPERAND = A**

The instruction format for immediate addressing mode is shown in Figure 6.2 below:

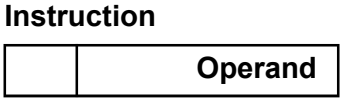



**Figure 6.2: Instruction Format for Immediate Addressing Mode**

In this mode, the source operand is a constant. The immediate data must be preceded by the "#" sign. It is very fast because the value to be loaded is included in the instruction. However, as the value is to be loaded is fixed at compile time, it is not very flexible.

In the figure 6.1, discussed in section 6.3, in the immediate mode of addressing the second word of the instruction is taken as the operand. i.e., the operand is 500.

This mode of addressing can be used to load information into any of the registers.

## 6.5 DIRECT ADDRESSING

When the operands are specified in the memory addressing mode, direct access to main memory data segment is required. However it results in slower processing of data. In direct addressing, the address field contains the effective address of the operand. The *Effective Address* is the location or the address where operand is present. It is defined to be the memory address obtained from the computation by the given addressing mode.

Direct addressing is so-named because the value to be referenced is obtained by specifying its memory address directly in the instruction.

***EA = A***

It requires only one memory reference and no special calculation. In the figure 6.1, discussed in section 6.3, the effective address is the address part of the instruction 500 and the operand to be loaded into AC is 800.

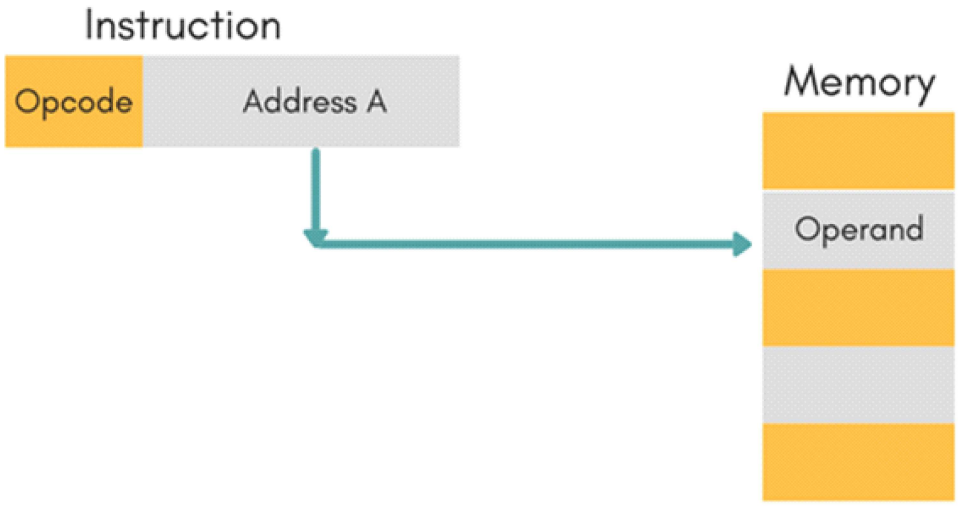



**Figure 6.3: Direct Addressing**

In direct memory addressing, one of the operands refers to a memory location and the other operand references a register.

Direct addressing is quite fast because although the value to be loaded is not included in the instruction, it is quickly accessible. It is also flexible than immediate addressing because the value to be loaded is given in the address.



**CHECK YOUR PROGRESS**

**Q.1:** Fill in the blanks:

i) The way in which the operand of an instruction is specified is referred to as .....

ii) In the ....., the operand is specified in the instruction itself.

iii) Immediate addressing is so-named because the value to be referenced immediately follows the ..... in the instruction.

iv) The addressing mode specifies a rule for interpreting the ..... field of an instruction before the ..... is actually executed.

v) The immediate data must be preceded by the ..... sign.

vi) The ..... is the location or the address where operand is present.

vii) ..... requires only one memory reference and no special calculation.

viii) In direct memory addressing, one of the operands refers to a ..... and the other operand references a .....

## 6.6 INDIRECT ADDRESSING

Indirect addressing is a powerful addressing mode that provides an exceptional level of flexibility. The limitation of direct addressing is that it provides only a limited address space as the length of the address field is usually less than the word length. To overcome this limitation, one solution can be to have the address field refer to the address of a word in memory, which in turn contains a full-length address of the operand. This is known as indirect addressing.

**EA = (A)**

The parentheses means "*Contents of*".

If the address bits of the instruction code are used as an address of the actual operand, it is termed as indirect addressing. Two memory references are needed to access data. First one is to access the address of the operand and the next one to access the data from that address.

The address field of instruction gives the address where the effective address is stored in memory. This slows down the execution, as this includes multiple memory lookups to find the operand.

In the figure 6.1 discussed in Section 6.3, the effective address is stored in memory at address 500. Therefore the effective address is 800 and the operand is 300.

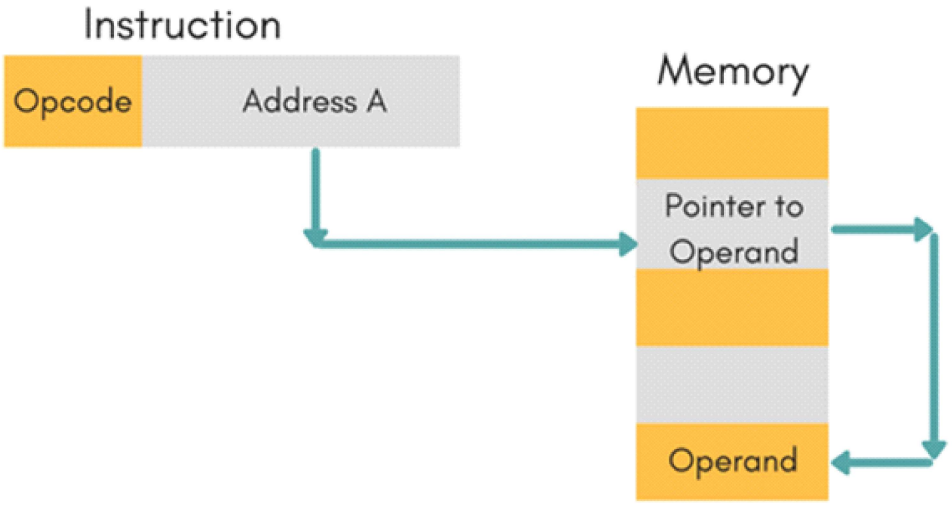



**Figure 6.4: Indirect Addressing**

The advantage of this addressing mode is that for a word length of N, an address space of $2^N$ is now available. However, if *k* is the length of the address field, the number of the effective addresses that may be referenced at any time is limited to $2^k$. The bits in the address field specify a memory address that is to be used as a pointer to the operand.

The disadvantage is that instruction execution requires two memory references to fetch the operand. One is to get its address and the second is to get its value.

## 6.7 REGISTER

Register addressing and direct addressing are almost similar with the only difference that the address field refers to a register in Register addressing rather than a main memory address.

**EA = R**

In register addressing mode, instead of memory a register is used to specify the operand. In the register mode, in the figure 6.1, discussed in section 6.3, the operand is in R1and 400 is loaded into AC. There is no effective address in this case.

The advantages of register addressing are that only a small address field is needed in the instruction and no time-consuming memory reference is required. The disadvantage of register addressing is that the address space is very limited. The access time for the registers is much lesser as compared to the main memory access. As there are only a limited number of registers as that of the main memory locations, so the use of the registers should be efficient. If the operand in the register that has been brought from the main memory is used for multiple operations, then the register addressing is used efficiently. However, if the operand in the register is used for only once in the processor and then after that it is returned back to the main memory, then it is a wastage step.

## 6.8 REGISTER INDIRECT

Register indirect addressing is also called indexed addressing or base addressing. To load a value from memory into a register using register-indirect addressing, a second register, known as the base register is used. This base register holds the actual memory address that the program is interested in. In this mode, the instruction specifies the register whose contents give us the address of operand which is in memory. Thus, the register contains the address of operand rather than the operand itself.

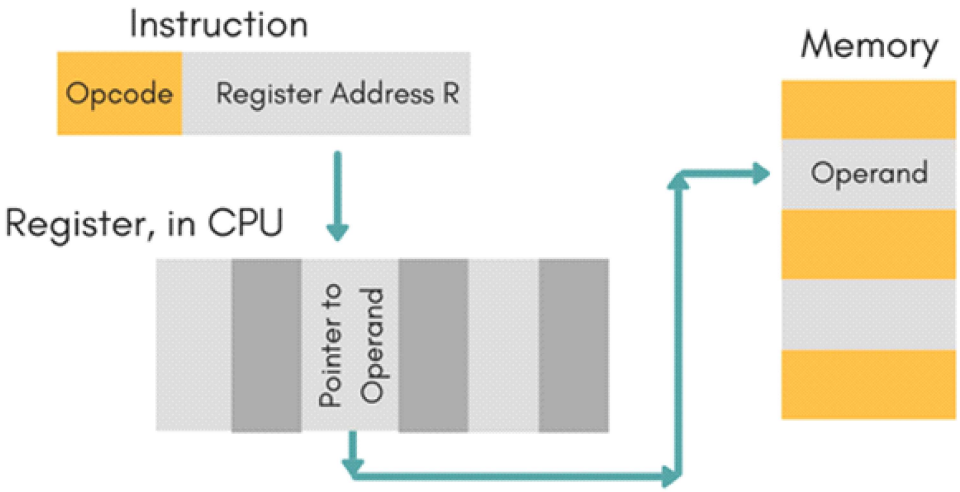



**Figure 6.5: Register Addressing**



## CHECK YOUR PROGRESS

**Q.2:** Fill in the blanks:

i) The techniques of specifying the address of the data are known as .....

ii) An instruction specifies the operation in the ..... field on the data specified by the operands in the operand field.

iii) ..... is a powerful addressing mode that provides an exceptional level of flexibility.

iv) The address field of instruction, in indirect mode of addressing, gives the address where the ..... is stored in memory.

v) The bits in the address field specify a memory address that is to be used as a ..... to the operand.

vi) The address field in Register addressing refers to a ..... rather than a main memory address.

vii) The memory access time for the registers is much ..... as compared to the main memory addresses.

viii) Register indirect addressing is also called ..... addressing.

ix) The ..... holds the actual memory address that the program is interested in.

x) The use of the registers in addressing should be ..... as there are a limited number of registers.



## 6.10 LET US SUM UP

- An instruction is a group of bits that instruct the computer to perform a specific operation.
- An instruction specifies the operation in the operation code field on the data specified by the operands in the operand field.
- An addressing mode can specify a location in memory, or a register or a constant.
- The techniques of specifying the address of the data are known as addressing modes.
- There are a variety of addressing techniques that are been applied. The most common of them are immediate, direct, indirect, register and register indirect.
- In the immediate mode, the operand is specified in the instruction itself.
- Immediate addressing is so-named because the value to be referenced immediately follows the operation code in the instruction.
- In direct addressing, the address field contains the effective address of the operand.
- *Effective Address* is defined to be the memory address obtained from the computation by the given addressing mode.
- In direct memory addressing, one of the operands refers to a memory location and the other operand references a register.
- If the address bits of the instruction code are used as an actual operand, it is termed as indirect addressing.
- Register addressing and direct addressing are almost similar with the only difference that the address field refers to a register in Register addressing rather than a main memory address.

- As there are only a limited number of registers as that of the main memory locations, so the use of the registers should be efficient.
- Register indirect addressing is also called indexed addressing or base addressing.
- In register indirect addressing the base register holds the actual memory address that the program is interested in.



## 6.11 FURTHER READING


1) Chaudhuri, P. Pal. (2nd Edition, 2003). *Compter Organization and Design*. PHI Learning Pvt. Ltd.
2) Gill, D. N. S. & Dixit, J. B. (2008). *Digital Design and Computer Organisation*. Firewall Media.
3) Null, L. & Lobur, J. (2014). *The Essentials of Computer Organization and Architecture*. Jones & Bartlett Publishers.
4) Stallings, William. (2004). *Computer Organization and Architecture Designing for Performance*. Pearson Education India.




## 6.12 ANSWERS TO CHECK YOUR PROGRESS

**Ans. to Q. No. 1:** i) Addressing; ii) immediate mode; iii) operation code; iv) address, operand; v) #; vi) Effective Address; vii) Direct addressing; viii) memory location, register

**Ans. to Q. No. 2:** i) addressing modes; ii) operation code; iii) Indirect addressing; iv) effective address; v) pointer; vi) register; vii) lesser; viii) indexed; ix) base register; x) efficient



## 6.13 MODEL QUESTIONS

**Q.1:** What is the purpose of using address mode techniques by a computer? Summarize the various addressing modes.

**Q.2:** What are the differences between direct and indirect addressing modes?

*** ***** ***